Plus courts chemins dans les graphes orientés

1 Plus courts chemins dans les graphes orientés

Nous avons développé en cours une théorie complète autour de la recherche de plus courts chemins dans les graphes orientés munis de fonctions longueurs à valeurs positives. Nous allons maintenant implémenter les différents aspects algorithmiques de cette théorie.

Pour disposer d'exemples, vous pourrez utiliser le module Python `graphes.py` livré avec le sujet. Il contient une fonction `graphe_oriente(n, k)` produisant un graphe sur n sommets. En variant le paramètre k, il est possible d'obtenir différents graphes de diverses densités d'arrêtes pour un nombre de sommets donné.

Il contient également une fonction `longueur(x, y)` qui associe une valeur numérique à chaque couple (x, y).

2 Routines utilitaires

1. Ecrivez une fonction Python `verification_chemin(G, c)` qui prend comme paramètres un graphe G et une liste c, et qui vérifie que la liste c représente bien un chemin du graphe G.

2. Ecrivez une fonction Python `verification_origine(c, o)` qui prend comme paramètres une liste c et un objet quelconque o, et qui vérifie que o est bien le premier élément de la liste c.

3. Ecrivez une fonction Python `verification_destination(c, d)` qui prend comme paramètres une liste c et un objet quelconque d, et qui vérifie que d est bien le dernier élément de la liste c.

4. Ecrivez une fonction Python `calcul_longueur(l, c)` qui prend comme paramètres une fonction longueur l prenant deux paramètres et retournant une valeur numérique, et une liste c, et qui calcule la longueur totale du chemin c pour la fonction longueur l.

3 Vérification des certificats

5. Ecrivez une fonction Python `verification_certificat_oui(G, l, c, s, t, k)` qui prend comme paramètres un graphe G, une fonction longueur l, une liste c, deux objets quelconques s et t, et un nombre k, et qui vérifie que c est bien un chemin de s à t dans G de longueur inférieure ou égale à k.

6. Ecrivez une fonction Python `verification_certificat_non(G, l, phi)` qui prend comme paramètres un graphe G, une fonction longueur l, une fonction `phi` à un paramètre et à valeur numérique, pouvant encoder un potentiel, et qui vérifie que `phi` est bien un potentiel pour le graphe G muni des longueurs l.

4 Algorithme de Dijkstra

7. Codez l'algorithme de Dijkstra.

8. Quelles vérifications pouvez-vous faire à l'issue de l'exécution de Dijkstra pour vous assurer que votre implémentation est correcte?

5 Version décision avec certificats

9. Implémentez l'algorithme pour la version décision du problème de plus courts chemins, avec certificats pour le OUI et pour le NON.

10. Comment pouvez-vous valider la bonne implémentation de cet algorithme?
